**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**отчет**

**по лабораторной работе №7**

**по дисциплине «Организация ЭВМ и систем»**

**Тема: Взаимодействие с внешними компонентами.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3388 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Титкова С.Д. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Молодцев Д.А. |

Санкт-Петербург

2024

**Цель работы**

Освоить программирование на ассемблере RISC-V взаимодействия с внешними устройствами (ввода и вывода).

**Задание:**

Для выполнения работы требуется реализовать ряд комбинационных функций на ассемблере. Каждое вычисленное значение функции определяет состояние светодиода: 0 – не горит, 1 – горит. Значение переменных определяется переключателями: 0 – выкл., 1 – вкл.

В задании требуется реализовать 3 функции для зажигания светодиодов. Порядок функций соответствует порядку светодиодов. Первый (самый левый) загорается красным, второй – зелёным, третий – синим. Также есть четвёртый светодиод, цвет которого – смесь всех цветов трёх других светодиодов. Так, если результат вычислений всех функций равен единице, то четвёртый светодиод горит белым цветом.

**Вариант 22.**

Номера и порядок функций - 7, 6, 9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | Y7 | Y6 | Y9 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Основные теоретические положения:**

Для созданных компонентов определяются адреса в памяти процессора, по которым осуществляется взаимодействие с компонентами. Адреса указаны в виде кода для языка С, который можно скопировать для программ. Пример:

// \* LED\_MATRIX\_0

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define LED\_MATRIX\_0\_BASE (0xf0000000)

#define LED\_MATRIX\_0\_SIZE (0x100)

#define LED\_MATRIX\_0\_WIDTH (0x8)

#define LED\_MATRIX\_0\_HEIGHT (0x8)

// \* SWITCHES\_0

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define SWITCHES\_0\_BASE (0xf0000110)

#define SWITCHES\_0\_SIZE (0x4)

#define SWITCHES\_0\_N (0x4)

Для всех компонентов определен Base адрес. Это – начальный адрес компонента. Для доступа к каждому последующему элементу требуется смещение на значение n, которое каждое для своего компонента. Так, для доступа к n-ному ключу требуется выполнить смещение n = x, где x – порядковый номер ключа при отсчете с нуля. Для светодиодной матрицы n = y\*W+x , где W – ширина матрицы, x,y – номера строки и столбца, на пересечении которых находится светодиод.

Для записи в память для установки цвета светодиода используется команда:

SW rs2, imm12(rs1)

Эта команда записывает слово (32 бита) по адресу rs1+imm12.

Получение информации об устройстве осуществляется чтением нужного участка памяти. В битах [16..23] расположен байт, определяющий выраженность красного цвета, в битах [8..15] - зеленого цвета, а в битах [0..7] (младший байт) - синего цвета.

Для чтения памяти используется команда:

LW rd, imm12(rs1)

Эта команда осуществляет чтение 32 бит памяти по адресу rs1+imm12 и записывает результат в регистр rd. Существует команда LWU с аналогичным назначением и применением для получения данных в беззнаковом виде.

**Выполнение работы**

Комбинационные функции были реализованы с помощью минимизации булевых функций методом карты Карно.

**Функция y7:**

Y7 = (x1 & x2)|(not(x1) & x4)|(not(x1) & not(x2) & x3) | (x3 & x4) |(x1 & not (x3) & not(x4))

Значение y7 = 0 или 1, 0 - красный светодиод выключен, 1 - включен.

**Функция y6:**

Y6 = (not(x1) & not(x2) & x4) | (x1 & x2 & x3 & x4) | (not(x1) & x3 & not(x4))

Значение y6 = 0 или 1, 0 - синий светодиод выключен, 1 - включен.

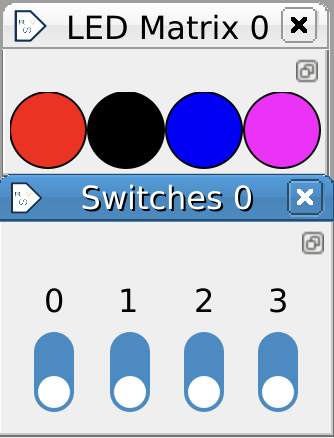
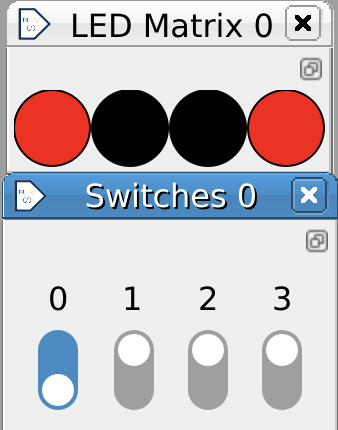
**Функция y9:**

Y9 = (not(x1) & x2 & not(x3)) | (not(x1) & x2 & x4) | (x1 & not(x2) & x4)

Значение y9 = 0 или 1, 0 - зелёный светодиод выключен, 1 - включен.

Исходный код разработанной программы представлен в приложении.

**Примеры выполнения показаны на рисунках**



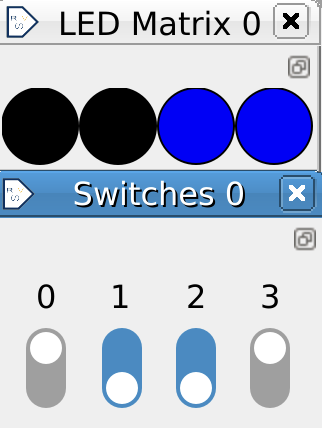
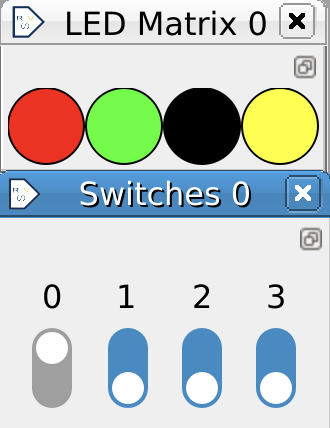


Рисунок 1

**Вывод**

В результате выполнения лабораторной работы были получены навыки работы с внешними устройствами. Также была изучена структура формирования цвета. Была разработана программа взаимодействия с внешними устройствами, такими как переключатели и светодиоды, на ассемблере RISC-V.

**ПРИЛОЖЕНИЕ**

**Код программы**

# запись начальных адресов компонентов

li a4 LED\_MATRIX\_0\_BASE

li a5 SWITCHES\_0\_BASE

# s0 - color[0]

# s1 - color[1]

# s2 - color[2]

# s3 - color[3]

# t0 - switch[0]

# t1 - switch[1]

# t2 - switch[2]

# t3 - switch[3]

# a6 - слово состояния ключей

loop: # бесконечный цикл

lw a6, 0[a5] # чтение состояния переключателя

# обнуление цветов

mv s0,zero

mv s1,zero

mv s2,zero

mv s3,zero

# перенос битов переключателя в отдельные регистры

andi t0,a6,1 #x1

andi t1,a6,2 #x2

andi t2,a6,4 #x3

andi t3,a6,8 #x4

# побитовый сдвиг для того

# чтобы привести состояние каждого переключателя к значению 0/1

srli t1,t1,1

srli t2,t2,2

srli t3,t3,3

# инвертирование состояний переключателей для будущего применения в функциях

not a0, t0

not a1, t1

not a2, t2

not a3, t3

jal y7\_function

beqz s8, skipColor0

lui s0,4080 # цвет светодиода 0 , s0 = 0x00FF0000 - RED

skipColor0:

jal y9\_function

beqz s8, skipColor1

li s1, 65280 # цвет светодиода 1, s1 = 0x0000FF00 - GREEN

skipColor1:

jal y6\_function

beqz s8, skipColor2

li s2, 255 # цвет светодиода 2, s2 = 0x000000FF - BLUE

skipColor2:

# цвет светодиода 3

or s3, s3, s0

or s3, s3, s1

or s3, s3, s2

skipColor3:

# пишем цвета в память для окраса

sw s0, 0(a4)

sw s1, 4(a4)

sw s2, 8(a4)

sw s3, 12(a4)

j loop # бесконечный цикл

y7\_function:

# F = (x1 & x2)|(not(x1) & x4)|(not(x1) & not(x2) & x3) | (x3 & x4) |(x1 & not (x3) & not(x4))

and s4, t0,t1 # x1 & x2

and s5, a0, t3 # not (x1) & x4

and s6, a0, a2 # not(x1) & not (x3)

and s6, s6, t2 # not(x1) & not (x3) & x3

and s7, t2, t3 # x3 & x4

and s8, t0, a2 # x1 & not (x3)

and s8, s8, a3 # x1 & not (x3) & not(x4)

or s8, s8, s7

or s8, s8, s6

or s8, s8, s5

or s8, s8, s4

# s8 = F

ret

y6\_function:

# F = (not(x1) & not(x2) & x4) | (x1 & x2 & x3 & x4) | (not(x1) & x3 & not(x4))

and s4, a0, a1 #not(x1) & not(x2)

and s4, s4, t3 #not(x1) & not(x2) & x4

and s5, t0, t1 #x1 & x2

and s5, s5, t2 #x1 & x2 & x3

and s5, s5, t3 #x1 & x2 & x3 & x4

and s8, a0, t2 #not(x1) & x3

and s8, s8, a3 #not(x1) & x3 & not(x4)

or s8, s8, s5

or s8, s8, s4

# s8 = F

ret

y9\_function:

# F = (not(x1) & x2 & not(x3)) | (not(x1) & x2 & x4) | (x1 & not(x2) & x4)

and s4, a0, t1 # (not x1) & x2

and s4, s4, a2 # not(x1) & x2 & not(x3)

and s5, a0, t1 # (not x1) & x2

and s5, s5, t3 # (not x1) & x2 & x4

and s8, t0, a1 # x1 & (not x2)

and s8, s8, t3 # x2 & (not x2) & x4

or s8, s8, s5

or s8, s8, s4

# s8 = F

ret